

Date=24/08/2020

Lecture By=Shubham Joshi

Subject  $\Rightarrow$  DP problem solving

IN PREVIOUS LECTURE (QUICK RECAP) Date-21/08/2020	In Today's Lecture (Overview)
<a href="#"><math>\Rightarrow</math> Dynamic programming in python</a> <a href="#"><math>\Rightarrow</math> Program to Print fibonacci series using dynamic programming</a> <a href="#"><math>\Rightarrow</math> MCQs</a> <a href="#"><math>\Rightarrow</math> Questions for self practice // CC and Assignment for the day</a>	Only One Problem Was discussed In the lecture  <a href="#">Question <math>\Rightarrow</math> Maximum profit from sale of wines</a>  <a href="#">Mcqs</a>  <a href="#">Questions For Self Practice // Assignment And CC for the Day</a>

## Question $\Rightarrow$ Maximum profit from sale of wines

Given  $n$  wines in a row, with integers denoting the cost of each wine respectively. Each year you can sale the first or the last wine in the row. However, the price of wines increases over time. Let the initial profits from the wines be  $P_1, P_2, P_3 \dots P_n$ . On the  $Y$ th year, the profit from the  $i$ th wine will be  $Y \cdot P_i$ . For each year, your task is to print "beg" or "end" denoting whether first or last wine should be sold. Also, calculate the maximum profit from all the wines.

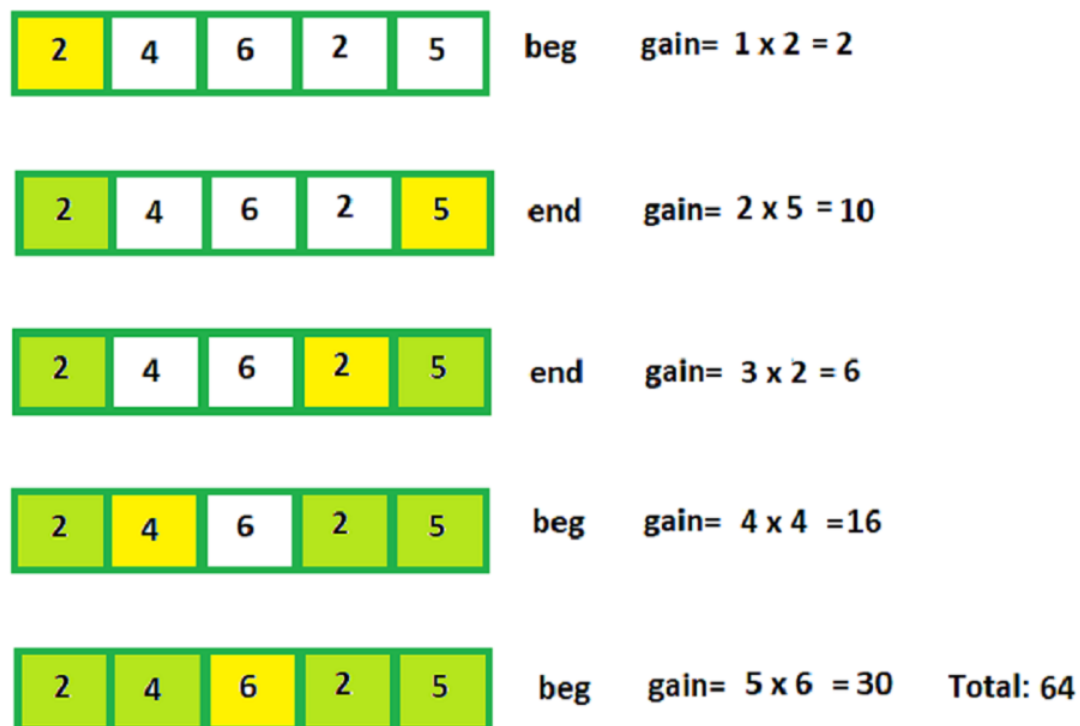
### Examples :

Input: Price of wines: 2 4 6 2 5

Output: beg end end beg beg

64

Explanation :



**Approach :** It is a standard Dynamic Programming problem. It initially looks like a greedy problem in which we should sell the cheaper of the wines each year but the example case (year 2) clearly proves the approach is wrong. Sometimes we need to sell an expensive wine earlier to save relatively costly wines for later years (Here, if 4 was sold in the 2nd year, in the 4th year we had to sell 2 which would be waste of a heavy coefficient).

The second problem is to “store the strategy” to obtain the calculated price which has a fairly standard method that can be used in other problems as well. The idea is to store the optimal action for each state and use that to navigate through the optimal states starting from the initial state.

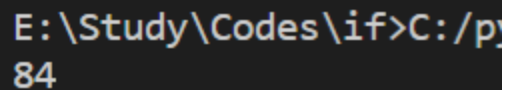
## Code

```
def solve(A,start,end,year):
    if start>end:
        return 0
    if dp[start][end][year]!=None:
        return dp[start][end][year]

    ans=
    max((A[start]*year)+solve(A,start+1,end,year+1),(A[end]*year)+solve(A,start,end-1,year+1))
    dp[start][end][year]=ans
    return ans

dp = [[[None for x in range(100)]for x in range(100)]for x in range(100)]
A=[2,4,5,6,7]
print(solve(A,0,len(A)-1,1))
```

## Output



```
E:\Study\Codes\if>C:/p
84
```

[This Code is From GeeksforGeeks <https://www.geeksforgeeks.org/maximum-profit-sale-wines/>]

## Mcqs

1.Which determines the time and space complexity for a dp ?

A=states

B=recursive calls stack

2.What n state dp what will be the size of memoization array ?

A=n

B=n + 1

C=n - 1

D=n / 2

## Questions For Self Practice // Assignment And CC for the Day

1.Maximum profit from sale of wines <https://www.geeksforgeeks.org/maximum-profit-sale-wines/>

2.Try to solve this with dp

<https://leetcode.com/problems/coin-change/> Q2.

<https://leetcode.com/problems/fibonacci-number/>